UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/726,902	12/03/2003	Mitchell Alsup	5500-88700	4177

53806 7590 11/06/2008

MEYERTONS, HOOD, KIVLIN, KOWERT & GOETZEL (AMD)
P.O. BOX 398
AUSTIN, TX 78767-0398

EXAMINER	
FENNEMA, ROBERT E	

ART UNIT	PAPER NUMBER
2183	

MAIL DATE	DELIVERY MODE
11/06/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 10/726,902
Filing Date: December 03, 2003
Appellant(s): ALSUP ET AL.

Robert C. Kowert (Reg. No. 39,255)
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 9/2/2008 appealing from the Office action mailed 3/31/2008

.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is incorrect. A correct statement of the status of the claims is as follows:

Appellant has remarked that the current claims are finally rejected, however, the Examiner re-opened the case, due to a new grounds of rejection, and at the time of the filing of the Appeal Brief, the claims stood rejected non-finally.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

6247121	**Akkary et al.**	**06-2001**
3896419	**Lange et al.**	**07-1975**
20040143721	**Pickett et al.**	**07-2004**

Rotenberg et al. "Trace Cache: A Low Latency Approach to High Bandwidth Instruction Fetching". Published in the Proceedings of the 29th Annual International Symposium on Microarchitecture, Dec 2-4, 1996. Pages 24-35.

Braught, Grant. "Class #21 - Assemblers, Labels & Pseudo Instructions". November 16, 2000.

Patterson, David. Hennessy, John. "Computer Architecture: A Quantitative Approach". Morgan Kaufmann Publishers, Inc. 2nd Edition, 1996. Pages 271-278.

Xia, Huaxia. "Using Trace Cache in SMT". June 10, 2001.

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 1-2, 11-15, 24-26, and 32-34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg et al. (herein Rotenberg), in view of Xia.

3. As per Claim 1, Rotenberg teaches: A microprocessor (Abstract), comprising:

an instruction cache configured to store instructions (Section 2.1, first paragraph);

a branch prediction unit (Section 2.1, first paragraph);

a trace cache configured to store a plurality of traces of instructions (Section 2.2, second paragraph); and

a prefetch unit coupled to the instruction cache, the branch prediction unit, and the trace cache (Figure 4, and Section 2.2. The Instruction latch appears to fill the role of a prefetch unit);

wherein the prefetch unit is configured to fetch instructions from the instruction cache until the branch prediction unit outputs a predicted target address (Section 2.2, where it is said that "on a trace cache miss, fetching proceeds normally from the instruction cache); and

wherein the prefetch unit is configured to check the trace cache for a match for the predicted target address in response to the branch prediction unit outputting the predicted target address (inherent in Rotenberg); and

wherein in response to the prefetch unit identifying a match for the predicted target address in the trace cache, the prefetch unit is configured to fetch one or more of the plurality of traces from the trace cache (Section 2.2, where it is said that "on a trace

cache hit, an entire trace of instructions is fed into the instruction latch, bypassing the instruction cache), but fails to teach:

wherein the prefetch unit is configured to not check the trace cache for a match until the branch prediction unit outputs the predicted target address.

Rotenberg teaches starting a trace on a trace cache miss, which then causes the line-fill buffer to begin filling (Section 2.2, fourth paragraph). Rotenberg does not specifically teach that an instruction must be a branch to access the cache (despite requiring branch prediction bits). However, Xia teaches that an advantage of starting a trace only at predictable branch instructions (Section 5.5, first scheme) is that less cache space is required, in direct contrast to Rotenberg, which teaches tracing on all instructions, requiring significantly more space (second scheme). Given this advantage, one of ordinary skill in the art at the time the invention was made would have been motivated by the advantage of needing less space for the trace cache to implement traces only on branches. In addition, because traces would only thus be started on branches, it wouldn't make sense to look for a hit in the trace cache on non-branches, since it couldn't possibly result in a hit, thus one of ordinary skill in the art would not search a trace cache when it is not required, and save power by not having to constantly access the cache with no hope of a hit.

4. As per Claim 2, Rotenberg teaches: The microprocessor of claim 1, wherein the branch prediction unit is configured to output the predicted target address in response to a prediction that a branch will be taken (Section 2.1, which discloses “They (the BTB

banks) serve the role of detecting branches in the instructions currently being fetched and providing their target addresses, in time for the next fetch cycle. A few paragraphs further down, it is stated that this happens "if there is a predicted taken branch").

5. As per Claim 11, Rotenberg teaches: The microprocessor of claim 1, wherein each of the plurality of traces comprises partially-decoded instructions (it is inherent that a trace comprises a partially-decoded instruction, otherwise the necessary control information as show in section 2.2 would not be available).

6. As per Claim 12, Rotenberg teaches: The microprocessor of claim 1, wherein each of the plurality of traces is associated with a tag comprising the address of an earliest instruction, in program order, stored within that trace (Section 2.2, where the tag identifies the starting address of the trace).

7. As per Claim 13, Rotenberg teaches: The microprocessor of claim 1, wherein each of the plurality of traces is associated with a flow control field comprising a label for an instruction to which control will pass for each branch operation comprised in that trace (Section 2.2, the branch flags exist for every branch in the trace, and indicate which instruction control will pass to (the taken or not taken)).

8. As per Claim 14, Rotenberg teaches: A computer system, comprising:

a system memory (inherent if the system has an instruction cache); and

a microprocessor coupled to the system memory (also inherent in Rotenberg's invention), comprising:

an instruction cache configured to store instructions (Section 2.1, first paragraph);

a branch prediction unit (Section 2.1, first paragraph);

a trace cache configured to store a plurality of traces of instructions (Section 2.2); and

a prefetch unit coupled to the instruction cache, the branch prediction unit, and the trace cache (Figure 4, and Section 2.2. The Instruction latch appears to fill the role of a prefetch unit);

wherein the prefetch unit is configured to fetch instructions from the instruction cache until the branch prediction unit outputs a predicted target address (Section 2.2, where it is said that "on a trace cache miss, fetching proceeds normally from the instruction cache);

wherein the prefetch unit is configured to check the trace cache for a match for the predicted target address in response to the branch prediction unit outputting the predicted target address (inherent in Rotenberg); and

wherein in response to the prefetch unit identifying a match for the predicted target address in the trace cache, the prefetch unit is configured to fetch one or more of the plurality of traces from the trace cache (Section 2.2, where it is said that "on a trace cache hit, an entire trace of instructions is fed into the instruction latch, bypassing the instruction cache), but fails to teach:

wherein the prefetch unit is configured to not check the trace cache for a match until the branch prediction unit outputs the predicted target address.

Rotenberg teaches starting a trace on a trace cache miss, which then causes the line-fill buffer to begin filling (Section 2.2, fourth paragraph). Rotenberg does not specifically teach that an instruction must be a branch to access the cache (despite requiring branch prediction bits). However, Xia teaches that an advantage of starting a trace only at predictable branch instructions (Section 5.5, first scheme) is that less cache space is required, in direct contrast to Rotenberg, which teaches tracing on all instructions, requiring significantly more space (second scheme). Given this advantage, one of ordinary skill in the art at the time the invention was made would have been motivated by the advantage of needing less space for the trace cache to implement traces only on branches. In addition, because traces would only thus be started on branches, it wouldn't make sense to look for a hit in the trace cache on non-branches, since it couldn't possibly result in a hit, thus one of ordinary skill in the art would not search a trace cache when it is not required, and save power by not having to constantly access the cache with no hope of a hit.

9. As per Claim 15, Rotenberg teaches: The computer system of claim 14, wherein the branch prediction unit is configured to output the predicted target address in response to a prediction that a branch will be taken (Section 2.1, which discloses "They (the BTB banks) serve the role of detecting branches in the instructions currently being fetched and providing their target addresses, in time for the next fetch cycle. A few

paragraphs further down, it is stated that this happens "if there is a predicted taken branch").

10. As per Claim 24, Rotenberg teaches: The computer system of claim 14, wherein each of the plurality of traces comprises partially-decoded instructions (it is inherent that a trace comprises a partially-decoded instruction, otherwise the necessary control information as show in section 2.2 would not be available).

11. As per Claim 25, Rotenberg teaches: The computer system of claim 14, wherein each of the plurality of traces is associated with a tag comprising the address of an earliest instruction, in program order, stored within that trace (Section 2.2, where the tag identifies the starting address of the trace).

12. As per Claim 26, Rotenberg teaches: The computer system of claim 14, wherein each of the plurality of traces is associated with a flow control field comprising a label for an instruction to which control will pass for each branch operation comprised in that trace (Section 2.2, the branch flags exist for every branch in the trace, and indicate which instruction control will pass to (the taken or not taken)).

13. As per Claim 32, Rotenberg teaches: A method, comprising:

fetching instructions from an instruction cache (Section 2.1, paragraphs 1-3);

continuing to fetch instructions from the instruction cache until a branch target address is generated (Section 2.2, where it is said that "on a trace cache miss, fetching proceeds normally from the instruction cache);

in response to a branch target address being generated, searching trace cache for an entry corresponding to the branch target address (Section 2.2, third paragraph), but fails to teach:

without searching a trace cache;

Rotenberg teaches starting a trace on a trace cache miss, which then causes the line-fill buffer to begin filling (Section 2.2, fourth paragraph). Rotenberg does not teach not searching the trace cache until a branch target address is generated, and in fact teaches that the trace cache is searched on every instruction. However, Xia teaches that an advantage of starting a trace only at predictable branch instructions (Section 5.5, first scheme) is that less cache space is required, in direct contrast to Rotenberg, which teaches tracing on all instructions, requiring significantly more space (second scheme). Given this advantage of using less cache space, one of ordinary skill in the art at the time the invention was made would have been motivated by the advantage of needing less space for the trace cache to implement traces only on branches. An effect of only having traces start on branches is that it is then inherent that a non-branch instruction can never be the start of a trace, therefore there would be no reason to search the trace cache for a non-branch instruction, which has further potential advantages such as power saving and potential critical path reduction, as one of ordinary skill in the art

would recognize.

14. As per Claim 33, Rotenberg teaches: The method of claim 32, further comprising continuing to fetch instructions from the instruction cache in response to no entry being identified in the trace cache corresponding to the branch target address (Section 2.2, where it is said that "on a trace cache miss, fetching proceeds normally from the instruction cache).

15. As per Claim 34, Rotenberg teaches: The method of claim 32, further comprising fetching one or more traces from the trace cache in response to an entry being identified in the trace cache corresponding to the branch target address (Section 2.2, where it is said that "on a trace cache hit, an entire trace of instructions is fed into the instruction latch, bypassing the instruction cache).

16. Claims 3, 16 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg and Xia, in view of Patterson et al. (herein Patterson).

17. As per Claim 3, Rotenberg teaches the microprocessor of claim 1, but fails to teach: wherein the branch prediction unit is configured to output the predicted target address in response to detection of a branch misprediction. However, Patterson teaches that in order to reduce the penalty for a misprediction, you can fetch both the taken and not taken instructions, and put them in the BTB, which would then be able to

immediately output the correct path on a misprediction without having to fetch it (Pages 276-277). While it would increase the cost of the system, the advantage is a smaller misprediction penalty, which may worth the cost, depending on the needs of the system. Therefore, one of ordinary skill in the art at the time the invention was made would have stored both the taken and not taken branch paths in the BTB in order to reduce the misprediction penalty, and thus increasing performance (by allowing the other path to be immediately output when the misprediction is noted).

18. As per Claim 16, Rotenberg teaches the computer system of claim 14, but fails to teach: wherein the branch prediction unit is configured to output the predicted target address in response to detection of a branch misprediction. However, Patterson teaches that in order to reduce the penalty for a misprediction, you can fetch both the taken and not taken instructions, and put them in the BTB, which would then be able to immediately output the correct path on a misprediction without having to fetch it (Pages 276-277). While it would increase the cost of the system, the advantage is a smaller misprediction penalty, which may worth the cost, depending on the needs of the system. Therefore, one of ordinary skill in the art at the time the invention was made would have stored both the taken and not taken branch paths in the BTB in order to reduce the misprediction penalty, and thus increasing performance (by allowing the other path to be immediately output when the misprediction is noted).

19. Claims 4, 10, 17, 23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg and Xia, in view of Braught, further in view of Xia.

20. As per Claim 4, Rotenberg teaches: The microprocessor of claim 1, further comprising a trace generator (it is necessary for Rotenberg's invention to have a trace generator in order to create traces), but fails to teach:

wherein the trace generator is configured to begin a trace with an instruction corresponding to a label boundary.

Braught teaches that in Assembly Language, most branches operate on labels, which the machine can only interpret when converted to an address (Page 3). In Braughts example on page 3, all branches are labels, making every branch a label boundary. When combined with Rotenberg's invention, this would mean that a trace would be generated with an instruction corresponding to a label boundary, as all the branches would go to labels, and even an address may be interpreted as a label. Given this knowledge, it would have been obvious to one of ordinary skill in the art that Rotenberg's invention begins traces when it encounters an instruction with a label boundary, as it only begins on branch instructions, which branch to labels.

21. As per Claim 10, Rotenberg teaches: The microprocessor of claim 4, wherein the trace generator is configured to generate traces in response to instructions being decoded (Section 2.2, the trace can not be completed (written into the cache) until the trace target addresses are calculated, which requires the instructions to be decoded).

22. As per Claim 17, Rotenberg teaches: The computer system of claim 14, further comprising a trace generator (it is necessary for Rotenberg's invention to have a trace generator in order to create traces), but fails to teach:

wherein the trace generator is configured to begin a trace with an instruction corresponding to a label boundary.

Braught teaches that in Assembly Language, most branches operate on labels, which the machine can only interpret when converted to an address (Page 3). In Braughts example on page 3, all branches are labels, making every branch a label boundary. When combined with Rotenberg's invention, this would mean that a trace would be generated with an instruction corresponding to a label boundary, as all the branches would go to labels, and even an address may be interpreted as a label. Given this knowledge, it would have been obvious to one of ordinary skill in the art that Rotenberg's invention begins traces when it encounters an instruction with a label boundary, as it only begins on branch instructions, which branch to labels.

23. As per Claim 23, Rotenberg teaches: The computer system of claim 17, wherein the trace generator is configured to generate traces in response to instructions being decoded (Section 2.2, the trace can not be completed (written into the cache) until the trace target addresses are calculated, which requires the instructions to be decoded).

24. Claims 5-8 and 18-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg, Xia and Braught, further in view of Lange et al. (USPN 3,896,419, herein Lange).

25. As per Claim 5, Rotenberg, Xia and Braught teach the microprocessor of claim 4, but fail to teach:

wherein the trace generator is configured to check the trace cache for a duplicate copy of the trace that the trace generator is constructing.

Rotenberg teaches a standard trace cache, with the potential for some duplication, when considering the alternative embodiment of path associativity disclosed in Section 2.3. And while Rotenberg does not discuss the issue of duplication, he does teach of disadvantages which occur when a trace cache miss occurs while servicing a previous trace cache miss, and that this situation needs to be avoided, otherwise features which increase cost or decrease performance must be implemented, and further teaches the disadvantages of a useless trace displacing a useful trace (Section 2.3, judicious trace selection). Therefore, Rotenberg teaches a system in which allows tracing of potential duplicate traces (because in order to trace multiple paths, every branch must be traced, even if it is a duplicate at the time, because it may not be a duplicate for future branches), and also a system which requires action when a miss occurs while servicing a miss. Lange teaches a system in which a cache is checked for a value while accessing memory, and if the identical value is found in the cache, the fetch to memory is aborted, freeing the memory to be used by a different operation

(preventing a blocking load, Column 9, Lines 60-65) This is analogous to Rotenberg in the sense that Lange provides a motivation to look for duplicates in a cache to prevent a large delay and allow another operation to continue (analogous to finding a solution to what to do when encountering a miss while servicing a miss) by implementing a solution of searching for the duplicate, and not performing a long-latency operation on the duplicate value since it is not required (if the previous trace appears to be a duplicate of what is in the trace, there is no need to continue tracing, and the duplicate value can be discarded as taught by Lange (there is no need to perform an action on a duplicate value), preventing the line-fill buffer from blocking new traces, a long-latency event).

In an alternative viewpoint disclosed above, Rotenberg also teaches why it is disadvantageous to displace a useful trace with a useless trace (under the interpretation that a duplicate trace being stored in the trace cache is useless, as the original can always be accessed in its place), thus, with the potential for duplicate traces to exist with path associativity in Rotenberg's alternative embodiments, Rotenberg indicates in his judicial trace selection that storing a duplicate trace would be at best useless, and at worst displace a useful trace that may be used, providing a motivation to prevent these duplicate traces from entering the trace cache from the line fill buffer, further providing a motivation to find some method to handle these cases, which Lange provides, by simply discarding the duplicate value.

Given these advantages in both of these situations, one of ordinary skill in the art at the time the invention was made would have been motivated to combine the teachings of Lange's checking for duplicates and discarding of the duplicates to

Rotenberg, in order to solve either or both of the problems of getting a miss while servicing a miss, and preventing eviction of a useful trace by a duplicate trace.

26. As per Claim 6, Lange teaches: The microprocessor of claim 5, wherein in response to the trace generator identifying a duplicate copy of the trace, the trace generator is configured to discard the trace under construction (Column 5, Lines 5-10).

27. As per Claim 7, Rotenberg teaches: The microprocessor of claim 5, wherein in response to the trace generator identifying an entry corresponding to a duplicate copy of the trace, the trace generator is configured to check the trace cache for an entry corresponding to a next trace to be generated (Section 2.2. The trade cache is checked every time there is a potential new trace, so when one trace is found and discarded, the next potential new trace will cause the trace cache to be checked again).

28. As per Claim 8, Lange teaches: The microprocessor of claim 7, wherein in response to the trace generator identifying a trace entry corresponding to the next trace to be generated, the trace generator is configured to discard the trace under construction (Column 5, Lines 5-10).

29. As per Claim 18, Rotenberg, Xia and Braught teach the computer system of claim 17, but fail to teach:

wherein the trace generator is configured to check the trace cache for a duplicate copy of the trace that the trace generator is constructing.

Rotenberg teaches a standard trace cache, with the potential for some duplication, when considering the alternative embodiment of path associativity disclosed in Section 2.3. And while Rotenberg does not discuss the issue of duplication, he does teach of disadvantages which occur when a trace cache miss occurs while servicing a previous trace cache miss, and that this situation needs to be avoided, otherwise features which increase cost or decrease performance must be implemented, and further teaches the disadvantages of a useless trace displacing a useful trace (Section 2.3, judicious trace selection). Therefore, Rotenberg teaches a system in which allows tracing of potential duplicate traces (because in order to trace multiple paths, every branch must be traced, even if it is a duplicate at the time, because it may not be a duplicate for future branches), and also a system which requires action when a miss occurs while servicing a miss. Lange teaches a system in which a cache is checked for a value while accessing memory, and if the identical value is found in the cache, the fetch to memory is aborted, freeing the memory to be used by a different operation (preventing a blocking load, Column 9, Lines 60-65) This is analogous to Rotenberg in the sense that Lange provides a motivation to look for duplicates in a cache to prevent a large delay and allow another operation to continue (analogous to finding a solution to what to do when encountering a miss while servicing a miss) by implementing a solution of searching for the duplicate, and not performing a long-latency operation on the duplicate value since it is not required (if the previous trace appears to be a duplicate of

what is in the trace, there is no need to continue tracing, and the duplicate value can be discarded as taught by Lange (there is no need to perform an action on a duplicate value), preventing the line-fill buffer from blocking new traces, a long-latency event).

In an alternative viewpoint disclosed above, Rotenberg also teaches why it is disadvantageous to displace a useful trace with a useless trace (under the interpretation that a duplicate trace being stored in the trace cache is useless, as the original can always be accessed in its place), thus, with the potential for duplicate traces to exist with path associativity in Rotenberg's alternative embodiments, Rotenberg indicates in his judicial trace selection that storing a duplicate trace would be at best useless, and at worst displace a useful trace that may be used, providing a motivation to prevent these duplicate traces from entering the trace cache from the line fill buffer, further providing a motivation to find some method to handle these cases, which Lange provides, by simply discarding the duplicate value.

Given these advantages in both of these situations, one of ordinary skill in the art at the time the invention was made would have been motivated to combine the teachings of Lange's checking for duplicates and discarding of the duplicates to Rotenberg, in order to solve either or both of the problems of getting a miss while servicing a miss, and preventing eviction of a useful trace by a duplicate trace.

30. As per Claim 19, Lange teaches: The computer system of claim 18, wherein in response to the trace generator identifying a duplicate copy of the trace, the trace

generator is configured to discard the trace under construction (Column 5, Lines 5-10).

31. As per Claim 20, Rotenberg teaches: The computer system of claim 18, wherein in response to the trace generator identifying an entry corresponding to a duplicate copy of the trace, the trace generator is configured to check the trace cache for an entry corresponding to a next trace to be generated (Section 2.2. The trade cache is checked every time there is a potential new trace, so when one trace is found and discarded, the next potential new trace will cause the trace cache to be checked again).

32. As per Claim 21, Lange teaches: The computer system of claim 20, wherein in response to the trace generator identifying a trace entry corresponding to the next trace to be generated, the trace generator is configured to discard the trace under construction (Column 5, Lines 5-10).

33. Claims 9 and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg, Xia, and Braught, in view of Akkary et al. (USPN 6,247,121, herein Akkary).

34. As per Claim 9, Rotenberg teaches the microprocessor of claim 4, but fails to teach:

wherein the trace generator is configured to generate traces in response to instructions being retired.

While Rotenberg teaches that the instructions need to be decoded (Section 2.2) before the trace can be generated, it is not taught that they need to be retired beforehand (although it is suggested in Section 2.3, fill issues). Akkary teaches that instructions are not put into the trace buffers until they have been retired, to ensure that they executed correctly (Column 3, Lines 40-44). Rotenberg discusses in Section 2.3 that some traces are committed but never used, thus evicting a useful trace. By ensuring that the trace is correct, the odds that it will be useful is increased, as an incorrect (not actually executed) trace should probably not need to be used, as branches tend to behave the same way (Section 1.1). Therefore, one of ordinary skill in the art at the time the invention was made would have waiting until an instruction was retired before considering adding it to the trace, in order to ensure accuracy of the trace and to prevent displacement of trace more likely to be reused.

35. As per Claim 22, Rotenberg teaches the computer system of claim 17, but fails to teach:

wherein the trace generator is configured to generate traces in response to instructions being retired.

While Rotenberg teaches that the instructions need to be decoded (Section 2.2) before the trace can be generated, it is not taught that they need to be retired beforehand (although it is suggested in Section 2.3, fill issues). Akkary teaches that instructions are not put into the trace buffers until they have been retired, to ensure that they executed correctly (Column 3, Lines 40-44). Rotenberg discusses in Section 2.3

that some traces are committed but never used, thus evicting a useful trace. By ensuring that the trace is correct, the odds that it will be useful is increased, as an incorrect (not actually executed) trace should probably not need to be used, as branches tend to behave the same way (Section 1.1). Therefore, one of ordinary skill in the art at the time the invention was made would have waiting until an instruction was retired before considering adding it to the trace, in order to ensure accuracy of the trace and to prevent displacement of trace more likely to be reused.

36. Claims 27-31 and 35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg, Xia, Braught, and Lange, further in view of Akkary.

37. As per Claim 27, Rotenberg teaches: A method, comprising:

beginning construction of a new trace (Section 2.2), but fails to teach:

receiving a retired instruction; and

determining if a previous trace under construction duplicates a trace in a trace cache and if the received instruction corresponds to a branch label; and

beginning construction of the trace in response to determining that a previous trace under construction duplicates a trace in a trace cache and that the received instruction corresponds to a branch label.

Rotenberg teaches starting a trace on a trace cache miss, which then causes the line-fill buffer to begin filling (Section 2.2, fourth paragraph). Rotenberg does not specifically teach that an instruction must be a branch to access the cache (despite

requiring branch prediction bits). However, Xia teaches that an advantage of starting a trace only at predictable branch instructions (Section 5.5, first scheme) is that less cache space is required, in direct contrast to Rotenberg, which teaches tracing on all instructions, requiring significantly more space (second scheme). Given this advantage, one of ordinary skill in the art at the time the invention was made would have been motivated by the advantage of needing less space for the trace cache to implement traces only on branches.

However, this combination still fails to teach starting a trace on a label boundary. Braught teaches that in Assembly Language, most branches operate on labels, which the machine can only interpret when converted to an address (Page 3). In Braughts example on page 3, all branches are labels, making every branch a label boundary. When combined with Rotenberg's invention, this would mean that a trace would be generated with an instruction corresponding to a label boundary, as all the branches would go to labels, and even an address may be interpreted as a label. Given this knowledge, it would have been obvious to one of ordinary skill in the art that Rotenberg's invention begins traces when it encounters an instruction with a label boundary, as it only begins on branch instructions, which branch to labels.

While Rotenberg teaches that the instructions need to be decoded (Section 2.2) before the trace can be generated, it is not taught that they need to be retired beforehand (although it is suggested in Section 2.3, fill issues). Akkary teaches that instructions are not put into the trace buffers until they have been retired, to ensure that they executed correctly (Column 3, Lines 40-44). Rotenberg discusses in Section 2.3

that some traces are committed but never used, thus evicting a useful trace. By ensuring that the trace is correct, the odds that it will be useful is increased, as an incorrect (not actually executed) trace should probably not need to be used, as branches tend to behave the same way (Section 1.1). Therefore, one of ordinary skill in the art at the time the invention was made would have waiting until an instruction was retired before considering adding it to the trace, in order to ensure accuracy of the trace and to prevent displacement of trace more likely to be reused.

Rotenberg teaches a standard trace cache, with the potential for some duplication, when considering the alternative embodiment of path associativity disclosed in Section 2.3. And while Rotenberg does not discuss the issue of duplication, he does teach of disadvantages which occur when a trace cache miss occurs while servicing a previous trace cache miss, and that this situation needs to be avoided, otherwise features which increase cost or decrease performance must be implemented, and further teaches the disadvantages of a useless trace displacing a useful trace (Section 2.3, judicious trace selection). Therefore, Rotenberg teaches a system in which allows tracing of potential duplicate traces, and also a system which requires action when a miss occurs while servicing a miss. Lange teaches a system in which a cache is checked for a value while accessing memory, and if the identical value is found in the cache, the fetch to memory is aborted, freeing the memory to be used by a different operation (preventing a blocking load, Column 9, Lines 60-65) This is analogous to Rotenberg in the sense that Lange provides a motivation to look for duplicates in a cache to prevent a large delay and allow another operation to continue (analogous to

finding a solution to what to do when encountering a miss while servicing a miss) by implementing a solution of searching for the duplicate, and not performing a long-latency operation on the duplicate value since it is not required (if the previous trace appears to be a duplicate of what is in the trace, there is no need to continue tracing, and the duplicate value can be discarded as taught by Lange (there is no need to perform an action on a duplicate value), preventing the line-fill buffer from blocking new traces, a long-latency event).

In an alternative viewpoint disclosed above, Rotenberg also teaches why it is disadvantageous to displace a useful trace with a useless trace (under the interpretation that a duplicate trace being stored in the trace cache is useless, as the original can always be accessed in its place), thus, with the potential for duplicate traces to exist with path associativity in Rotenberg's alternative embodiments, Rotenberg indicates in his judicial trace selection that storing a duplicate trace would be at best useless, and at worst displace a useful trace that may be used, providing a motivation to prevent these duplicate traces from entering the trace cache from the line fill buffer, further providing a motivation to find some method to handle these cases, which Lange provides, by simply discarding the duplicate value.

Given these advantages in both of these situations, one of ordinary skill in the art at the time the invention was made would have been motivated to combine the teachings of Lange's checking for duplicates and discarding of the duplicates to Rotenberg, in order to solve either or both of the problems of getting a miss while servicing a miss, and preventing eviction of a useful trace by a duplicate trace.

The fact that traces can only begin on branch labels means that the next trace cannot be started until an instruction represents a branch label, regardless of when the duplication is found and dealt with.

38. As per claim 28, Rotenberg teaches: The method of claim 27, further comprising continuing construction of an incomplete trace already in process in response to determining that the incomplete trace does not duplicate a trace in the trace cache (Section 2.2).

39. As per Claim 29, Lange teaches: The method of claim 27, further comprising searching the trace cache for duplicate entries subsequent to completion of the previous trace under construction or the new trace (Column 5, Lines 5-10).

40. As per Claim 30, Rotenberg teaches: The method of claim 29, further comprising creating a new entry in the trace cache in response to no duplicate entry being identified (Section 2.2).

41. As per Claim 31, Lange teaches: The method of claim 29, further comprising discarding a trace in response to a duplicate entry being identified (Column 5, Lines 5-10).

42. As per Claim 35, Rotenberg teaches: A microprocessor comprising:

means for starting a new trace (Section 2.2), but fails to teach:

means for receiving a retired operation;

means for determining if a previous trace under construction duplicates a trace in a trace cache and if the received operation is a first operation at a branch label; and

means for starting a new trace in response to determining that a previous trace under construction duplicates a trace in trace cache and that the received operation is a first operation at a branch label.

Rotenberg teaches starting a trace on a trace cache miss, which then causes the line-fill buffer to begin filling (Section 2.2, fourth paragraph). Rotenberg does not specifically teach that an instruction must be a branch to access the cache (despite requiring branch prediction bits). However, Xia teaches that an advantage of starting a trace only at predictable branch instructions (Section 5.5, first scheme) is that less cache space is required, in direct contrast to Rotenberg, which teaches tracing on all instructions, requiring significantly more space (second scheme). Given this advantage, one of ordinary skill in the art at the time the invention was made would have been motivated by the advantage of needing less space for the trace cache to implement traces only on branches.

However, this combination still fails to teach starting a trace on a label boundary. Braught teaches that in Assembly Language, most branches operate on labels, which the machine can only interpret when converted to an address (Page 3). In Braughts example on page 3, all branches are labels, making every branch a label boundary. When combined with Rotenberg's invention, this would mean that a trace would be

generated with an instruction corresponding to a label boundary, as all the branches would go to labels, and even an address may be interpreted as a label. Given this knowledge, it would have been obvious to one of ordinary skill in the art that Rotenberg's invention begins traces when it encounters an instruction with a label boundary, as it only begins on branch instructions, which branch to labels.

While Rotenberg teaches that the instructions need to be decoded (Section 2.2) before the trace can be generated, it is not taught that they need to be retired beforehand (although it is suggested in Section 2.3, fill issues). Akkary teaches that instructions are not put into the trace buffers until they have been retired, to ensure that they executed correctly (Column 3, Lines 40-44). Rotenberg discusses in Section 2.3 that some traces are committed but never used, thus evicting a useful trace. By ensuring that the trace is correct, the odds that it will be useful is increased, as an incorrect (not actually executed) trace should probably not need to be used, as branches tend to behave the same way (Section 1.1). Therefore, one of ordinary skill in the art at the time the invention was made would have waiting until an instruction was retired before considering adding it to the trace, in order to ensure accuracy of the trace and to prevent displacement of trace more likely to be reused.

Rotenberg teaches a standard trace cache, with the potential for some duplication, when considering the alternative embodiment of path associativity disclosed in Section 2.3. And while Rotenberg does not discuss the issue of duplication, he does teach of disadvantages which occur when a trace cache miss occurs while servicing a previous trace cache miss, and that this situation needs to be avoided, otherwise

features which increase cost or decrease performance must be implemented, and further teaches the disadvantages of a useless trace displacing a useful trace (Section 2.3, judicious trace selection). Therefore, Rotenberg teaches a system in which allows tracing of potential duplicate traces, and also a system which requires action when a miss occurs while servicing a miss. Lange teaches a system in which a cache is checked for a value while accessing memory, and if the identical value is found in the cache, the fetch to memory is aborted, freeing the memory to be used by a different operation (preventing a blocking load, Column 9, Lines 60-65) This is analogous to Rotenberg in the sense that Lange provides a motivation to look for duplicates in a cache to prevent a large delay and allow another operation to continue (analogous to finding a solution to what to do when encountering a miss while servicing a miss) by implementing a solution of searching for the duplicate, and not performing a long-latency operation on the duplicate value since it is not required (if the previous trace appears to be a duplicate of what is in the trace, there is no need to continue tracing, and the duplicate value can be discarded as taught by Lange (there is no need to perform an action on a duplicate value), preventing the line-fill buffer from blocking new traces, a long-latency event).

In an alternative viewpoint disclosed above, Rotenberg also teaches why it is disadvantageous to displace a useful trace with a useless trace (under the interpretation that a duplicate trace being stored in the trace cache is useless, as the original can always be accessed in its place), thus, with the potential for duplicate traces to exist with path associativity in Rotenberg's alternative embodiments, Rotenberg indicates in

his judicial trace selection that storing a duplicate trace would be at best useless, and at worst displace a useful trace that may be used, providing a motivation to prevent these duplicate traces from entering the trace cache from the line fill buffer, further providing a motivation to find some method to handle these cases, which Lange provides, by simply discarding the duplicate value.

Given these advantages in both of these situations, one of ordinary skill in the art at the time the invention was made would have been motivated to combine the teachings of Lange's checking for duplicates and discarding of the duplicates to Rotenberg, in order to solve either or both of the problems of getting a miss while servicing a miss, and preventing eviction of a useful trace by a duplicate trace.

The fact that traces can only begin on branch labels means that the next trace cannot be started until an instruction represents a branch label, regardless of when the duplication is found and dealt with.

(10) Response to Argument

43. Regarding Appellant's arguments for Claims 1-2, 13, 14-15, 25, and 32-34, on Pages 11-13, Appellant appears to be misinterpreting the Examiners rejection of the claim limitation involving "wherein the prefetch unit is configured to not check the trace cache for a match for the predicted target address in response to the branch prediction unit outputting the predicted target address". Examiner has argued that it is impossible to check for a match of a predicted target address, until that predicted target address

has been generated. Appellant continues to cite the next limitation when referring to the Examiners argument for this limitation, instead of addressing this limitation by itself. Examiner has admited in the rejection that Rotenberg, by itself, checks the trace cache every cycle, for something, however, Appellant is disregarding the Examiners argument that you cannot check for a predicted target address until it is generate, because it is fundamentally impossible to check for a value before that value is known or generated. Therefore, Examiner believes that the Appellants arguments in Section 1 are moot, as the Appellant appears to be arguing that the Examiners rejection for one limitation does not apply to another, which Examiner has not attempted to argue, and Examiner has already agreed that Rotenberg does not teach the limitation the Appellant argues in Section 1 that Rotenberg does not teach.

44. On Pages 13 and 14, in Section 2, Appellant now directly addresses the limitation of not checking the trace cache for a match until the branch prediction unit outputs the predicted target address, and argues that Xia teaches checking the trace cache in parallel. However, Appellant has completely ignored the Examiners actual rejection, which points to Section 5.5 of Xia, which shows an advantage for only starting traces on branches. Additionally, Appellant has ignored the Examiners rejection where the Examiner has stated that given this information, one of ordinary skill in the art would clearly recognize that if traces can only start on branches, and you do not have a branch instruction, there is absolutely no chance that you could possibly have a hit in the trace cache, and therefore, it would be wasteful to even attempt to search the cache

for something that is guaranteed to not be there. One of ordinary skill in the art would recognize that using any computer component requires both power and time, both of which computer designers desperately attempt to minimize in all implementations. One of ordinary skill in the art, recognizing that there is zero chance for any cache hit on a non branch instruction, would clearly be motivated to not check a cache for a value that is impossible to be found, when there are well known advantages for not doing so. Appellant appears to be bodily incorporating Rotenberg into Xia, and ignoring what the art teaches to one of ordinary skill in the art. The test for obviousness is not whether the features of a secondary reference may be bodily incorporated into the structure of the primary reference, nor is it that the claim invention must be expressly suggested in any one or all of the references. Rather, the test is what the combined teachings of the references would have suggested to those of ordinary skill in the art. See *In re Keller,* 642 F.2d 413, 208 USPQ 871 (CCPA 1981).

By that standard, the Examiner has laid out a proper rejection, which shows what one of ordinary skill in the art would have taken away from the combination of references, while the Appellant has argued each reference individually, and has attempted to bodily incorporate the two, instead of recognizing that Xia is an improvement over Rotenberg, and applying the references as such, in accordance with the law set forth above.

45. In Section 3 of the Appellant's arguments, the Appellant has argued that the Examiner has not provided a proper reason to combine the references, however,

Appellant has conceded in this section that Xia taught advantages over Rotenberg, and Examiner notes that Xia explicitly discloses Rotenberg as a reference in Section 2, and in their references section, therefore, Examiner believes that an extremely solid case for combination is present in the case. Appellant further argues against the Examiners assertion regarding power saving or critical path reduction, however, Examiner reminds the Appellant that the test for obviousness is not what is explicitly suggested in the reference, but rather, it is what the combined teachings of the references would have suggested to those of ordinary skill in the art, and both power and time savings are basic, fundamental issues that one of ordinary skill in the art would clearly recognize, as laid out by the Examiner earlier.

46. In regards to Appellant's Section 4, while Examiner has explained how the combination teaches the claimed invention before, he will do so again at this point for clarity. Rotenberg teaches checking the cache for a hit on every instruction. Xia, as the Appellant has pointed out, also, in its most basic implementation, does the same. However, as Appellant has ignored in his remarks, Xia teaches that an optimization of the cache (Section 5.5) is that the trace lines start from branches. The advantage of this, by itself, is that the cache miss rate can be reduced, a significant time saving feature. That is what is expressly suggested in the references. However, one must not only look at what is expressly taught, but what is suggested to one of ordinary skill in the art. If one knows that a trace can only start on a branch, then one would also recognize that a trace cannot start on a non-branch. Therefore, it is a certainty that if we were to

check the cache on a non-branch, we would miss in the cache, which Xia teaches, and one of ordinary skill in the art would recognize, is an extreme disadvantage. One of ordinary skill in the art would then recognize, if missing in a cache not only incurs a large time penalty due to the miss, but also wastes power (one of skill in the art would recognize that a circuit in use draws power, a circuit not in use does not draw power), and one of the most fundamental improvements in computers over the last several decades has been the desire to use less power. Therefore, what this suggests to one of ordinary skill in the art is that if they know that there is absolutely no chance to hit in the cache, and missing in the cache incurs great penalty, it is clearly in their best interest to not check the cache. There are multiple advantages for doing so, and multiple disadvantages if they go ahead and check it, therefore, Examiner asserts that the combination of references suggest, to one of ordinary skill in the art, the claimed invention, as Examiner has laid out in the Office Action.

47. Regarding Appellants arguments on Page 17, to Claims 11 and 24, Appellants have argued that the assertion that the trace cache inherently holds "partially-decoded" instructions is incorrect, however, section 2.2 of Rotenberg reference clearly shows that the information stored with the trace contains things such as the next and starting address of the trace, which is not information that can be gained without at least "partially decoding" the instruction, since an instruction is just a random collection of bits until it is decoded and put into a proper context. Appellant has argued that the control information is "not part of the instruction traces themselves", however, Examiner does

not agree with this assertion, Appellant appears to be arbitrarily declaring what is and isn't part of the trace, Examiner asserts that if data is gathered which is required for the trace to work properly, and was gained by decoding the instructions, then it is a partially-decoded part of the instruction trace. The very nature of a trace cache requires a partial decoding, and for Appellant to suggest that this is incorrect is a misunderstanding of the art, and incorrect.

48. Regarding Appellants arguments regarding Claims 13, 26, 4, and 17, Appellant has argued that the trace target address and trace fall-through address are not labels for each branch operation comprised in the trace. However, the branch flags in every single instruction trace indicate in which direction each and every branch in the trace goes. Additionally, given the lack of a solid definition for "label", Examiner asserts that the rejection is proper.

49. Regarding Appellants arguments for Claims 3 and 16, Appellant has argued that the limitation of "wherein the branch prediction unit is configured to output the predicted target address in response to detection of a branch misprediction is not taught by the combination of references. However, the combination proposed would have the branch predictor hold both the taken and not taken paths, therefore, on a misprediction, the other path is available to be output immediately, therefore, teaching the limitation. Appellant has argued that Claim 3 does not teach a branch predictor, however, this is immaterial, as the branch predictor is not the issue, the fact that the combination of

references teaches the limitation is the issue that the Appellant needs to address.

50. Regarding Appellants arguments for Claims 10 and 23, Appellant has argued that Rotenberg does not teach "generating traces in response to instructions being decoded", but as described in previous sections, the trace requires data from decoded instructions in order to make the trace, thus, the traces are generated in response to the decoding, even if it is not an immediate response.

51. Regarding Appellants arguments for Claims 5 and 18, Appellant has argued that Rotenberg does not teach "wherein the trace generator is configured to check the trace cache for a duplicate copy of the trace that is being constructed", and has argued that the reference does not allow for duplicate traces, thus they will never check for them. However, the Examiner has pointed out that there are alternate embodiments which can allow for duplicate traces, explicitly indicate in Rotenberg, Section 2.3, where there are multiple paths, thus the same trace under construction could be a duplicate of a trace in the trace cache, until the time it is complete (since the multiple paths may not diverge until the last instruction, or never, a trace could be a duplicate for substantially the entire trace, until the end), thus the potential for duplicates exist, because if you trace multiple paths, you must trace everything, meaning the potential for duplicates arise, if there is no divergence, as the Examiner has explained above. The art combined with Rotenberg details the inefficiencies of duplicates, and why it is advantageous to check for them, and make sure they do not occur. The Appellant has refused to address the possibility

of this alternate embodiment, which is explicitly indicated in the primary reference, thus, the Appellants arguments are not directed at the actual rejection of the claim, but rather, are based on the incorrect situation of duplicate paths not potentially existing, and thus, are improper. Again, much like the independent claims, The Appellant is ignoring the case law which states that one must not look at what is expressly suggested in the references, but rather, what one of ordinary skill in the art would take away from the teachings of the reference.

52. Regarding Appellants arguments for Claims 6-7 and 19-20, the Appellants arguments are essentially repeats of arguments presented earlier, and Examiner believes his remarks for the previous claims address the arguments presented for these claims.

53. Regarding Claims 8 and 21, Appellants have argued that the combination of references does not teach discarding a trace under construction in response to identifying a trace entry corresponding to the next trace to be generated. Again, Examiner asserts that the Appellant is attempting to bodily incorporate the references, instead of looking at what they teach to one of ordinary skill in the art. Lange teaches that there are extreme disadvantages to storing duplicate trace copies, therefore, if one is discovered, one of ordinary skill in the art would clearly be motivated to discard the duplicate trace if it is discovered. Examiner has clearly laid out a motivation to combine

the references in the rejection, despite Appellants claim to the contrary that "Examiner has not provided any motivation or other reason to combine the references".

54. Regarding Claims 9 and 22, Appellant has argued that the references teach the opposite of the claim; however, Appellant is once again attempting bodily incorporation, instead of reading what is taught to one of ordinary skill in the art. The combination of references teaches that retired instructions are guaranteed to be correct, and that making a trace of incorrect instructions would cause useful traces to be evicted from the cache. Examiner believes that one of ordinary skill in the art, knowing these two pieces of information, would clearly see that a trace of retired instructions will be less likely to displace a useful cache line, and therefore, clearly be advantageous.

55. Appellants remaining arguments are all substantially identical to arguments previously presented, and Examiner refers to his previous remarks in order to address these remarks, for brevity's sake.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Robert Fennema

/Robert Fennema/

Conferees:

Eddie Chan

/Eddie P Chan/

Supervisory Patent Examiner, Art Unit 2183

Kevin Ellis
/Kevin L Ellis/
Acting SPE of Art Unit 2187